Compilateur JavaSketchpad®

Manuel d'utilisation

Avertissement

Ceci est le manuel du compilateur en ligne et non la documentation de JavaSketchpad. Pour celle-ci se reporter au site de Key Curriculum Press : http://www.keypress.com/sketchpad/javasketchpad/

La syntaxe est celle du compilateur en ligne et non la syntaxe d'origine de JavaSketchpad, les différences étant notées au passage. Pour une interface graphique, voir Key Curriculum Press.

Table des matières

Syntaxe générale p.2
- Commentaires p.2
- Lignes de code p.3
- Options p.4

Points p.4
Droites p.5
Cercles p.5
Surfaces p.6
Mesures p.6
Calculs p.7
Transformations p.8
Textes p.9
Lieu p.9
Boutons p.10
Constructions et exemples p.11
- Arcs p.11
- Intersections p.11
- Constructions conditionnelles p.12
- Utilisation de calculs p.13

Import/Export p.14
Annexes p.15
- Options d'applet p.15
- Sommaire des entités p.16
- Fonctions non disponibles p.17
- Macros p.18
- Copyright et licences p.19

Syntaxe générale

Un dessin JavaSketchpad se compose de l'applet Java et de la description du dessin.
Dans une page HTML, ceci est défini comme :

```
<applet code="GSP.class" codebase="../jsp" archive="jsp4.jar" width="800" height="500">
<param name=Frame value=0>
<param name=MeasureInDegrees value=0>
<param name=Construction value="
      …/…
">
affichage alternatif
</applet>
```

L'applet Java est définie par le fichier jsp4.jar
Le dessin lui même par la valeur (value) du paramètre 'Construction'
D'autres paramètres permettent de définir la couleur du fond etc. Voir en annexe.
Ces paramètres de l'applet ne sont pas modifiables en ligne par le compilateur.
L'affichage alternatif est, en HTML, ce qui est affiché à la place de l'applet si le moteur Java est indisponible ou désactivé.

Chaque ligne du Construction.value de l'applet est conforme à la syntaxe d'origine de JavaSketchpad, telle que définie par Key Curriculum Press. C'est le résultat de l'exécution du compilateur à partir du texte source, qui, lui, obéit à une syntaxe différente.
Seul le format du texte source est défini ici, bien qu'il soit très proche de la syntaxe d'origine, il comporte des différences importantes.

Commentaires

Les lignes vides sont ignorées.
Un commentaire est une ligne mise entre accolades { }
Les commentaires de plus d'une ligne doivent avoir sur chaque ligne leur deux accolades.
Les caractères accolades, et les caractères " sont interdits dans le texte du commentaire.

Exemples:

```
{       *** Ceci est un commentaire ***       }
{ et ceci est un commentaire }
{ sur plusieurs lignes }
          { les espaces en dehors des accolades sont ignorés }
```

Ligne de code

Une ligne de code se présente sous la forme générale

{ <id> **}** <entité> **(** <paramètres> …)…(<paramètres> …**) [**<options>…**] ;**

et est terminée par un point virgule **;**

<id> est l'identifiant de l'entité géométrique, son nom en fait.
Il est composé de chiffres lettres ou _@§&~|!?=:/^*+.– ou espaces et ne doit pas commencer par un chiffre ou un espace. Les caractères # et $ sont réservés.

Exemples :
{A1} {A*} {A^B} {A - B} {@ABC}

Les chiffres et espaces avant, ainsi que les espaces après sont ignorés, ainsi
{1 A} , { 1A } ou { A } définit le même nom A que {A}

{ id } est facultatif, en son absence un identifiant est généré automatiquement par le compilateur.

<entité> est le type d'entité géométrique généré : Point, Segment, Circle etc.

<paramètres> Les listes de paramètres, séparées par des virgules et entre parenthèses sont définies selon l'entité, par exemple les coordonnées du point, le centre du cercle, le texte d'un message etc. :

```
Point (100,200);
Circle (A, B);
FixedText (20,40,'ceci est du texte');
ShowButton (400,40,'Montrer')(A,B);
```

Tous les paramètres doivent être définis avant leur utilisation. Les "références en avant" sont interdites : dans Circle (A, B); les points A et B doivent être définis au préalable.

L'id d'un élément peut être son rang dans la liste des éléments
`{A} Point (100, 200);`
`{B} Point (100, 300);`
`Segment (1,2);` est alors identique à `Segment (A,B)`, et est remplacé par `Segment (A,B)` dans le source à la compilation. Ceci permet d'importer des constructions existantes dans le compilateur.
Bien sûr dans tous les cas ce qui est généré en sortie est `Segment (1,2)`, seul compris par l'applet.

<options>
Entre crochets[ ], définissent les options d'affichage de l'élément, le label affiché, sa couleur etc. :

```
Point (100,200) [label('A'), red];
```

Nota :
1) Ne pas confondre le label (ce qui est affiché) et le nom (qui sert à identifier l'élément dans le script)

```
{A} Point (100,200)[label('B')];
```
A est le nom de l'élément, pour référence ultérieure par exemple dans un Segment (A,…)
B est ce qui est affiché sur le dessin.
Bien sûr c'est plus pratique de mettre le même, pour éviter des erreurs…

2) { } est en fait un commentaire pour JavaSketchpad lui-même. Ignoré par l'applet, il ne sert qu'au compilateur. Par ailleurs des commentaires après le ; sont autorisés dans l'applet, mais interdits dans le compilateur. De même plusieurs entités sur la même ligne sont interdites.

Point(100,200); { autre point } Point (150,300); est une ligne illégale

Options

label('text') définit le label affiché d'un point. Seuls les points peuvent avoir un label
le texte ne doit pas comporter de caractères '', ou ressemblant à des balises html
les ' doivent être doublés ' ', pour simuler '' on écrit donc ' ' ' '
black, white, red, green, blue, yellow, cyan, magenta couleurs prédéfinies
color(r, g, b) couleur arbitraire, r, g, b composantes rouge, vert, bleu de 0 à 255
hidden l'élément est invisible
thick trait épais, seuls les éléments 'linéaires' peuvent être épais
traced l'élément laisse une trace en se déplaçant, les élément surfaciques sont interdits
layer(n) couche de dessin, plus n est grand plus l'élément est dessus les autres n = 1 à 999

D'autres options, disponibles pour JavaSketchpad, ne sont pas traitées par le compilateur

Points de base

Les points peuvent être définis au départ, ou résultat de constructions géométriques.

Point (x,y); Définit un point déplaçable, coordonnées initiales x, y en pixels sur l'applet
x, y sont limités à la surface visible soit ici $0 \le x < 800$, $0 \le y < 500$
y est vers le bas, bien que l'orientation de l'axe des ordonnées soit vers le haut,
comme il se doit, le pixel 0 est en haut et le pixel 499 en bas !

FixedPoint (x,y); Définit un point non déplaçable par la souris.

Point on object (obj, k); Définit un point déplaçable, contraint à rester sur l'objet obj qui est
un segment, ray, line, circle ou polygon (pourtour).
k définit la position initiale (rapport d'un segment, angle sur le cercle…)

Exemple:

```
{AB} Segment (A,B);
Point on object (AB, 0.333333);
```
Le point est contraint à rester sur le segment AB, au départ il est au 1/3 du segment à partir de B (!!!)

Les autres sortes de points sont le résultat des constructions géométriques

Midpoint (segment); Milieu du segment

Intersect (straight, straight); Intersection de deux éléments rectilignes

Intersect1 (obj, circle); 1er point d'intersection avec le cercle, obj est un autre cercle ou un élément
rectiligne. Il est compliqué de savoir si le point que l'on veut est le 1er ou le 2ème
point d'intersection de deux cercles. Il est alors conseillé d'utiliser des constructions
alternatives de ce point.
Intersect2 (obj, circle); 2ème point d'intersection avec le cercle.

Points obtenus par transformation géométrique d'autres points : voir "transformations"

Droites

Les droites sont définies par deux points quelconques préalablement définis comme points de base ou comme points résultat de constructions préalables. Elles sont orientées point2 → point1.

Segment (pointB, pointA); Définit le segment AB

Ray (pointB, pointA); Définit la demi-droite AB, d'origine A (!!!)

Line (pointB, pointA); Définit la droite infinie AB

Parallel (straight, P); Parallèle à la droite/segment/1/2droite straight, passant par P
Orientée dans le même sens

Perpendicular (straight, P); Perpendiculaire à la droite/segment/1/2droite straight, passant par P
Orientée à 90° dans le sens horaire (donc rotation de $-\pi/2$)

Les autres droites sont obtenues comme transformations géométriques de segment, ray et line préalablement construits, en gardant leur type, un segment transformé en segment etc.
Voir 'transformations".

Cercles

Un cercle est défini par son centre et un point ou son rayon, résultats de points de base, de constructions ou de mesures préalables.

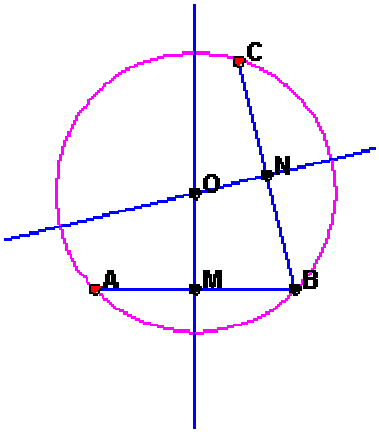
Circle (center, point); Cercle de centre donné passant par un point donné

Circle by radius (center, radius); Cercle de centre donné et de rayon donné.

Le rayon peut être un segment, une mesure de longueur ou le résultat d'un calcul. Le compilateur interdit un rayon qui n'est visiblement pas une longueur : surface, angle, slope ou ratio sont interdits. Les passer à travers un calcul, à vos risques et périls pour la signification géométrique douteuse qui peut en résulter.

Les cercles passant par trois points doivent être explicitement construits.
Le centre est le point d'intersection des médiatrices de deux des segments, perpendiculaires en leurs milieux. Donc :

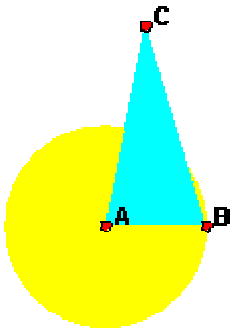
```
{AB} Segment (A,B);
{BC} Segment (B,C);
{M} Midpoint (AB);
{N} Midpoint (BC);
{d1} Perpendicular (AB, M);
{d2} Perpendicular(BC, N);
{O} Intersect (d1, d2);
Circle (O, A);
```



Les éléments intermédiaires peuvent être déclarés [hidden] pour ne pas surcharger le dessin.
Le compilateur ne gère actuellement pas de macros qui permettraient de définir tout cela comme
Circumcircle (A, B, C)

D'autres constructions au chapitre "constructions classiques".

Eléments surfaciques



Circle interior (circle); Définit le disque formé par l'intérieur du cercle. Il est entièrement rempli de la couleur par défaut ou indiquée.

Polygon (A,B,C…Z) Définit le polygone fermé décrit par les points successifs, dans l'ordre A, B, …Z. Au minimum 3 points., puis peint l'intérieur du polygone
Le contour peut être utilisé comme chemin pour les Point on object et les Locus

Le pourtour n'est pas dessiné. Utiliser un Circle ou une succession de Segment pour le visualiser.
Ces éléments surfaciques jouent facilement à cache-cache les uns derrière les autres et il est fortement conseillé de définir ceux qui sont devant avec l'option layer()
L'option thick n'a aucun sens et l'option traced remplirait facilement tout le dessin et est rejetée par le compilateur.

Ces éléments peuvent être copiés par les transformations géométriques.

Mesures et calculs

Les mesures permettent d'afficher les valeurs de longueurs angles surfaces mesurées sur le dessin, mais aussi peuvent être utilisées comme paramètres dans d'autres constructions.

Toutes comportent les paramètres :
x, y position de l'affichage de la mesure
'text' texte affiché devant la mesure
JavaSketchpad permet d'afficher un suffixe après la mesure, ceci n'est pas actuellement géré par le compilateur.

Si on veut les utiliser juste pour définir des paramètres dans les constructions ultérieures, on peut les déclarer [hidden].

Parameter (valeur, x, y, 'text'); Définit un paramètre constant égal à 'valeur'

Length (segment, x, y, 'text'); Longueur du segment, en pixels

Distance (A, B, x, y, 'text'); Distance entre les points A et B, en pixels

Perimeter (Poly, x, y, 'texte'); Périmètre du polygone

Circumference (circle, x, y, 'text'); Périmètre du cercle

Radius (circle, x, y, 'text'); Rayon du cercle, en pixels

Area (obj, x, y, 'text'); Aire en pixel² de l'objet cercle, polygone ou disque

Angle (A, B, C, x, y, 'text'); Angle (BA,BC), en radians entre $-\pi$ et $+\pi$
L'applet autorise au choix des angles affichés en radians ou en degrés, le compilateur place cette option à "radians" pour être compatible avec les autres valeurs d'angles qui sont toujours en radians. + si sens trigonométrique de A vers C, – si sens horaire

Slope (straight, x, y, 'text'); Pente de l'objet rectiligne (tangente de l'angle avec l'horizontale)

Ratio/Segments (sgmt1, sgmt2, x, y, 'text'); segm1/segm2 (!!!)

Ratio/Points (A, B, C, x, y, 'text'); AC/AB (!!!) préférer Ratio/Points à Ratio/Segment si des points peuvent se confondre, supprimant alors le segment du dessin.

Ces mesures sont typées par le compilateur, il est illégal d'injecter une aire ou un angle comme valeur de rayon etc. Types de valeurs :

Dist	longueurs, distances, périmètres
Angle	angles
Area	aires
Ratio	valeurs sans dimension : pente, rapports de longueurs
Calc	valeurs calculées ou paramètre

Calculs

Les mesures peuvent être injectées dans un calcul :

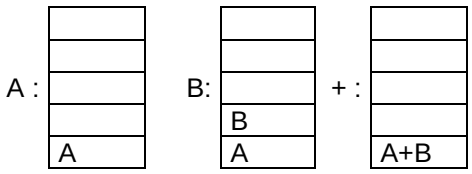
Calculate (x, y, 'text', 'expr') (p1, p2, … pn);

Ce calcul est fait à partir des paramètres p1 à pn qui sont des mesures ou calculs préalables.
expr est la description du calcul en notation Polonaise inverse.

Chaque élément de expr est
- Un nombre ≥ 0, avec éventuellement des espaces pour séparer deux nombres
- L'indice d'un paramètre : p1 est représenté par A, p2 par B etc. maximum de 26 paramètres donc.
- Une opération + – * / ! ou ^, ^ représentant l'élévation à une puissance, ! la négation
- Une fonction @xxxx parmi
  @sin_ @cos_ @tan_ @asin @acos @atan
  @sgn_ @abs_ @rond @trnc @sqrt @ln__ @log_

@ln__ est le logarithme neperien, @log_ le logarithme décimal
@rond arrondit à l'entier le plus proche
@trnc arrondit à l'entier inférieur en valeur absolue
@sgn_ -1, 0, 1
noter les **4** caractères après @, complétés par des '_'
Les angles sont exprimés en radians

La notation Polonaise inverse consiste à empiler les valeurs successives et à les consommer lors des opérations.
Ainsi A+B s'écrit 'AB+' en notation polonaise inverse :
Mettre A sur la pile, mettre B sur la pile, additionner les deux valeurs du sommet de la pile et mettre le résultat sur la pile à la place



Cette notation rend superflue toute parenthèse et l'applet Java exécute les opération directement en les lisant de gauche à droite.
'AB+C*' représente donc (A+B)*C que l'on peut écrire d'ailleurs aussi 'CAB+*'
puisque (A+B)*C = C*(A+B) enfin (A+B)*(C+D) s'écrit ainsi 'AB+CD+*'

Le résultat du calcul est le sommet de la pile à la fin. Il ne doit rien rester d'autre sur la pile.
Les opérations non commutatives / - et ^ donnent
'AB/' = A/B, 'AB –' = A – B et 'AB^' = A^B (A puissance B)

Les espaces sont superflus sauf pour séparer deux valeurs numériques consécutives
par exemple '1 2+'
L'ambiguïté de 'AB–1…' entre calculer A–B puis mettre +1 sur la pile ou mettre –1 sur la pile en plus de A et B est résolue par l'absence effective de – unaire, contrairement à ce qui est dit dans la doc de JavaSketchpad, –1 représente toujours – (opération) puis empiler 1. Les valeurs négatives sont obtenues par x! (exemple 3.5! : vaut -3.5)

Transformations géométriques

Les éléments déjà construits peuvent être copiés via une transformation géométrique.
La copie à le même type que l'objet d'origine, un segment donne un segment etc.

Reflection (obj, straight); L'objet 'obj' est copié par symétrie par rapport à la droite, segment ou ray 'straight'

Dilation (obj, P, k); Homothétie de centre P de rapport k

Dilation/SegmentRatio (obj, P, segm1, segm2); Homothétie de centre P de rapport segm1/segm2

Dilation/3PtRatio (obj, P, A, B, C); Homothétie de centre P de rapport AB/AC.
Même remarque que pour Ratio/points, préférer 3PtRatio à SegmentRatio si les points peuvent se confondre.

Dilation/MarkedRatio (obj, P, ratio); Homothétie de centre P, ratio défini par la mesure ou le calcul préalable 'ratio'

Rotation (obj, P, a); Rotation de centre P et d'angle a radians

Rotation/MarkedAngle (obj, P, A, B, C); Rotation de centre P et d'angle ABC

Rotation/MeasuredAngle (obj, P, angle); Rotation de centre P et d'angle mesuré au préalable

Translation (obj, dx, dy); Translation de l'objet de (dx, dy) (dy>0 vers le haut !!!)

VectorTranslation (obj, A, B) Translation selon le vecteur AB

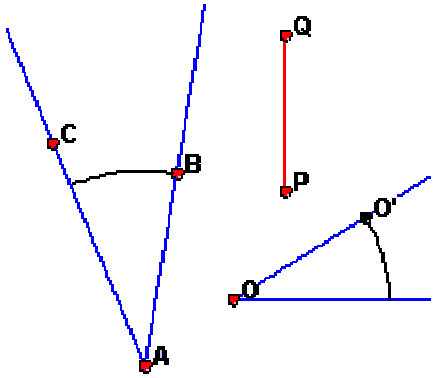
Translation/FixedAngle/MarkedDistance (obj, d, a_num)
Translation de la distance d mesurée au préalable, selon la direction de l'angle a radians avec l'horizontale

Translation/MarkedAngle/FixedDistance (obj, a, d_num)
Translation de la distance d pixels, selon la direction de l'angle a, mesuré au préalable, avec l'horizontale.

Translation/MarkedAngle/MarkedDistance (obj, a, d)
Translation de la distance d mesurée au préalable, selon la direction de l'angle a mesuré au préalable, avec l'horizontale.

```
{d} Distance (P,Q,10,10,' ')[hidden];
{a} Angle (B,A,C,10,10,' ')[hidden];

{O1} Translation/MarkedAngle/MarkedDistance (O,a,d)[label('O''')];
```

Déplace le point O dans la direction définie par l'angle BAC sur une distance égale à PQ



Textes

Outre les mesures, il est possible d'afficher des textes quelconques :

FixedText (x, y, 'texte'); Affiche le texte à la position x,y sur l'écran de l'applet
L'objet texte ainsi défini peut être copié par :

PeggedText (P, obj); L'objet texte ou la mesure obj est 'accroché' au point P, et se déplace donc avec le point.

ConcatText (x, y, obj, obj, …); Les textes/mesures obj sont concaténés et l'objet texte résultant est affiché à la position x,y.

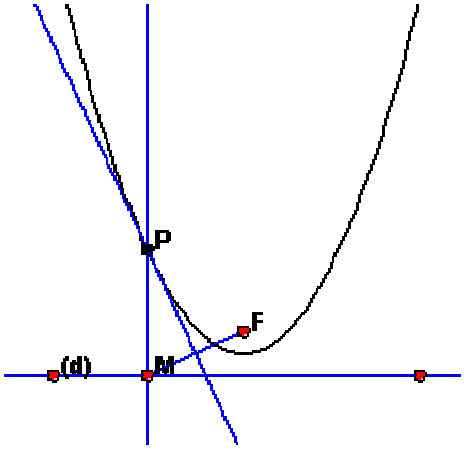
Lieu géométrique

Locus (P, M, chemin, n); Trace le lieu du point P quand le point libre M suit le trajet chemin ce trajet étant un segment, ray, line, circle ou polygon. M doit être déclaré comme étant déjà sur un certain objet ('Point on object'), et chemin doit être cet objet. n est le nombre de points calculés pour tracer le lieu.

Les lieux obtenus ne peuvent pas être copiés par une transformation et ne peuvent pas servir pour des autres "Point on object".
Aucune vérification n'est effectuée que le point P est bien une construction qui dépend de M.

Exemple : tracé d'une parabole

```
{A} Point (50,400)[label('(d)')];
{B} Point (600,400);
{AB} Line (A,B);
{F} Point (300,320)[label('F')];
{M} Point on object (AB, 0.8)[label('M')];
{FM} Segment (F,M);
{H} Midpoint (FM)[hidden];
{m} Perpendicular (FM,H);
{d} Perpendicular (AB,M);
{P} Intersect (m,d)[label('P')];
{parab} Locus (P,M,AB,100);
```



Le lieu ainsi tracé se déforme en temps réel quand on déplace les points de base A,B et F
Le déplacement manuel du point M permet de voir comment est parcourue la parabole

Le tracé du lieu d'un segment/ray/line est moins intéressant car il se contente d'afficher le segment pour chaque position calculée, l'enveloppe apparaît ainsi mais noyée dans un fouillis de lignes.
Ici l'enveloppe de m est la même parabole, `Locus (m,M,AB,100);` fait apparaître cette enveloppe mais trace 100 lignes m sur le dessin !

Lorsque le lieu d'un point est discontinu (asymptotes) des lignes parasites peuvent apparaître, si n est suffisamment grand, ces lignes parasites sont en fait … les asymptotes.

Boutons

Les boutons permettent des actions comme cacher/montrer des parties du dessin, ou animer automatiquement la construction.

Comme les mesures, ils comportent systématiquement leur position x,y et leur titre.

HideButton (x, y, 'titre')(obj, obj, …); Tous les objets indiqués sont cachés.

ShowButton (x, y, 'titre')(obj, obj, …); Tous les objets indiqués sont affichés, même si déclarés [hidden] à l'origine.

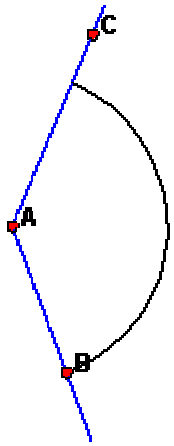
MoveButton (x, y, v, 'titre')(A, A', B, B', …); Les points libres A', B' … sont respectivement déplacés vers les points A, B … à la vitesse v pixels par trame. (attention : destination, point …)

AnimateButton (x,y,'titre') (A, Pa, B, Pb, …) (va, vb, …) (fa, fb, …) (ma, mb, …);
Les points libres A, B, … sont respectivement déplacés sur les chemins Pa, Pb, … aux vitesses va, vb, … un point déclaré 'on object' doit être déplacé sur cet objet.
fa, fb, … et ma, mb, … définissent leur façon de se déplacer
f = 0 répète indéfiniment, f = 1 une seule fois
m = 0 sens anti-horaire sur un cercle, aller-retour sur une droite
m = 1 sens horaire sur un cercle, aller simple sur une droite

SimultaneousButton (x,y,'titre')(obj, obj, …); Tous les boutons obj sont actionnés. Ceci permet de déclarer un {v} ShowButton [hidden] et un {h} HideButton [hidden],
un SimultaneousButton (v,h) provoque alors simultanément l'apparition des objets déclarés dans v et la disparition des objets déclarés dans h.

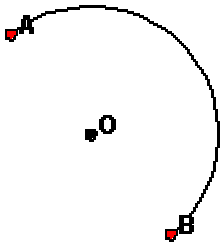
Constructions classiques et astuces

Tracé d'un arc




```
{ ================ }
{ *** Draw arc *** }
{ ================ }
{A} Point (100,200)[label('A')];
{B} Point (150,300)[label('B')];
{C} Point (130,100)[label('C')];
{BC} Segment (B,C)[hidden];
{M} Point on object (BC,0.3)[hidden];
{P} Rotation/MarkedAngle (B,A,B,A,M)[hidden];
{arc} Locus (P,M,BC,100);
{Ab} Ray (B,A);
{Ac} Ray (C,A);
```

L'arc généré n'est correct que si l'angle est suffisamment inférieur à π. Sinon il faut le découper en plusieurs arcs plus petits. En particulier pour tracer un demi-cercle.

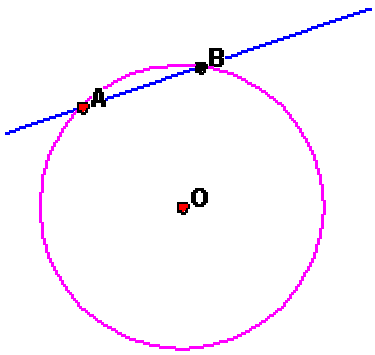



```
{ ======================= }
{ *** Draw half circle *** }
{ ======================= }
{A} Point (100,100)[label('A')];
{B} Point (180,200)[label('B')];
{AB} Segment (A,B)[hidden];
{O} Midpoint (AB)[label('O')];
{  pi/2  }
{H} Rotation (B,O, 1.5707963267948966)[hidden];
{s1} Segment (A,H)[hidden];
{M1} Point on object (s1, 0)[hidden];
{P1} Rotation/MarkedAngle (A,O,A,O,M1)[hidden];
{arc1} Locus (P1,M1,s1,100);
{s2} Segment (B,H)[hidden];
{M2} Point on object (s2, 0)[hidden];
{P2} Rotation/MarkedAngle (B,O,B,O,M2)[hidden];
{arc2} Locus (P2,M2,s2,100);
```

Pour tracer l'autre demi-cercle, remplacer π/2 par –π/2 ou échanger A et B
Le tracé d'un secteur ou d'un demi disque ne se fait pas aussi simplement : il faut effectivement tracer n sommets d' un polygone…

Autre intersection

Lorsqu'une droite d passant par A recoupe un cercle passant par A et de centre O, l'autre point d'intersection ne doit pas être cherché par Intersect1/2 car cela ne sera jamais le bon d'après la loi de Murphy : on obtiendra de nouveau A ! L'astuce est de le construire autrement :




```
{O} Point (100, 200)[label('O')];
{A} Point (50,150)[label('A')];
{D} Point (400, 30);
{d} Line (A,D);
{C} Circle (O,A);
{ *** deuxième intersection : *** }
{m} Perpendicular (d, O)[hidden];
{B} Reflection (A, m)[label('B')];
```

Et de même pour la deuxième intersection de deux cercles

Intersection d'une demi-droite

En fait le choix Intersect1/Intersect2 avec une droite peut être maîtrisé car il ne dépend que de la définition de la droite. La droite (A,B) est orientée de B vers A et Intersect1 est la première et Intersect2 la deuxième sur ce sens de parcours, donc dans ce sens Intersect1 est le point d'entrée dans le cercle et Intersect2 le point de sortie du cercle.

De même avec une demi droite Ray(A,B) d'origine B, orientée de B vers A et un Segment (A,B) orienté de B vers A.

Tracé d'une horizontale/verticale

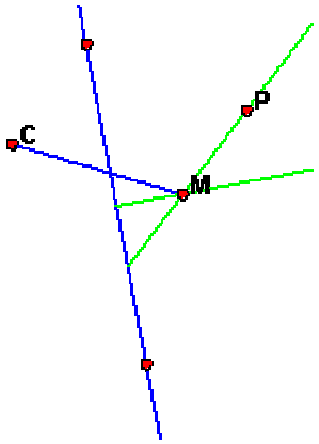
Il suffit de déplacer le point de base par translation de dx, 0 ou 0, dy

```
{ horizontale passant par A }
{A.} Translation (A,100,0)[hidden];
{d} Line (A,A.);
```

Constructions conditionnelles

Soit par exemple à restreindre un point M dans un demi plan, défini par une droite (d) et un point P
La perpendiculaire à (d) en M coupe (d) en I La droite PM coupe(d) en J
Les demi-droites IM et JP se coupent (en M) seulement si P et M du même côté de (d). La suite de la construction est alors à partir de ce point d'intersection au lieu de M lui même

```
{ droite (d) }
{A} Point (100,10);
{B} Point (150,400);
{d} Line (A,B);
{P} Point (200,100)[label('P')];
{M} Point (300,200);
{ test }
{q} Perpendicular (d,M)[hidden];
{I} Intersect(d,q)[hidden];
{PM} Line (P,M)[hidden];
{J} Intersect (d,PM)[hidden];
{Jp} Ray (P,J)[hidden];
{Im} Ray (M,I)[hidden];
{M.} Intersect(Jp, Im)[label('M'),red];

{ utilisation de M. }
{C} Point (160,300)[label('C')];
{CM.} Segment(C,M.);
```



(les points ont été déplacés pour l'illustration et les demi-droites rendues visibles en vert)
Le segment CM etc. n'existent que si P et M du même côté de (d) : M et CM disparaissent sinon ainsi que toutes les constructions en découlant.

Le même genre d'astuce (intersection avec des demi-droites ou des segments) peut être utilisée dans d'autres cas semblables pour s'assurer que ce que l'on construit respecte les conditions du problème. Inversement, on peut être amené à prolonger des segments en droites pour traiter tous les cas de figure.

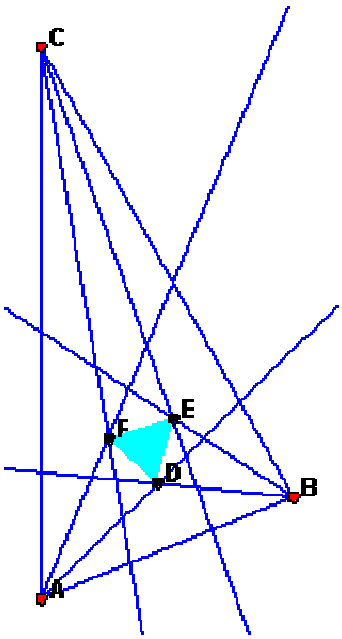
Une autre possibilité de dessin conditionnel ou complexe est l'usage judicieux de Calculate, et par exemple de @sgn_ ou @abs_ pour obtenir des valeurs nulles si condition.

Utilisation de calculs

Les constructions classiques à la règle et au compas s'obtiennent toutes à partir des droites et cercles tracés par les fonctions de base. La fonction **Calculate** permet d'aller au delà de ces constructions, construire un polygone régulier à 7 côtés, trisecter un angle quelconque etc.

Comme exemple citons le théorème de Morley qui nécessite de trisecter les angles d'un triangle.

```
{ *** triangle de Morley *** }
{ ************************* }

{A} Point (100,400)[label('A')];
{B} Point (400,400)[label('B')];
{C} Point (200,100)[label('C')];
{AB} Segment (A,B);
{BC} Segment (B,C);
{CA} Segment (C,A);

{ mesure l'angle A.. et calcule A/3.. }
{a} Angle (B,A,C,10,10,' ')[hidden];
{a/3} Calculate (10,10,' ','A3/')(a)[hidden];
{b} Angle (C,B,A,10,10,' ')[hidden];
{b/3} Calculate (10,10,' ','A3/')(b)[hidden];
{c} Angle (A,C,B,10,10,' ')[hidden];
{c/3} Calculate (10,10,' ','A3/')(c)[hidden];

{ tracer les trisectrices }
{B1} Rotation/MeasuredAngle (B,A,a/3)[hidden];
{AB1} Ray (B1,A);
{B2} Rotation/MeasuredAngle (B1,A,a/3)[hidden];
{AB2} Ray (B2,A);

{C1} Rotation/MeasuredAngle (C,B,b/3)[hidden];
{BC1} Ray (C1,B);
{C2} Rotation/MeasuredAngle (C1,B,b/3)[hidden];
{BC2} Ray (C2,B);

{A1} Rotation/MeasuredAngle (A,C,c/3)[hidden];
{CA1} Ray (A1,C);
{A2} Rotation/MeasuredAngle (A1,C,c/3)[hidden];
{CA2} Ray (A2,C);

{ triangle de Morley }
{D} Intersect (AB1,BC2)[label('D')];
{E} Intersect (BC1,CA2)[label('E')];
{F} Intersect (CA1,AB2)[label('F')];
{t} Polygon (D,E,F)[cyan];
```



Le triangle DEF est équilatéral, quel que soit le triangle ABC de départ.

Import/Export

L'import de source par copier/coller depuis un script existant vers la fenêtre "source" nécessite juste quelques précautions concernant les commentaires et les identifiants

S'assurer qu'il n'y a pas de commentaires imbriqués ni mélangés dans une ligne de code

Il est recommander de définir explicitement un identifiant pour chaque entité du dessin.
Le compilateur en génère une sinon, mais cet identifiant automatique peut conduire à des doubles définitions ultérieures en cas de modifications.

Enfin certaines constructions de JavaSketchpad sont inconnues du compilateur.

Le réimport en tant que source d'un script traduit par le compilateur ne pose aucun problème. Et est forcément compatible "par construction".

Le compilateur est conçu à l'origine pour générer en ligne une applet exécutée sur la même page que le compilateur, ou exportée automatiquement dans une page créée à la volée sur mon site.
Le script généré peut être exporté et placé dans vos propres pages/applications par copier-coller depuis la fenêtre "traduite" du compilateur, en le mettant dans une balise <applet> </applet> idoine.

Vous devez alors vous procurer l'applet JavaSketchpad auprès de Key Curriculum Press :
http://www.keypress.com/sketchpad/javasketchpad/ pour pouvoir l'exécuter sur vos pages.

Enfin le source peut être sauvegardé par copier-coller depuis la fenêtre "source" vers un fichier texte quelconque, mais il est de toute façon compatible avec le script traduit et peut être régénéré à partir de ce script traduit. (donc disponible dans le source des pages HTML générées)

Il est aussi possible d'utiliser le compilateur en ligne pour générer un dessin, que vous copiez en image statique par copie d'écran.

Annexes

Options d'applet

Bien que non modifiables par le compilateur en ligne, elles peuvent être redéfinies dans vos propres pages.

Couleur de fond : défaut = gris 200 200 200
<param name=BackRed value=0**>**
<param name=BackGreen value=255**>**
<param name=BackBlue value=0**>**

Encadré : 0 pas encadré, 1 encadré
<param name=Frame value=0**>**

Police pour les labels :
<param name=LabelFont value=…**>** défaut = Helvetica
<param name=LabelSize value=6…100**>** défaut = 12
<param name=LabelBold value=0/1**>** défaut = 1
<param name=LabelItalic value=0/1**>** défaut = 0

Police pour les boutons : idem **ActionFont**, **ActionSize**, **ActionBold**, **ActionItalic**
Défaut = TimesRoman 14 0 0

Police pour les mesure et textes : idem **MeasureFont, MeasureSize, MeasureBold, MeasureItalic**
Défaut = Helvetica 10 0 0 La taille est redéfinie à 14 pour le compilateur en ligne

Mesures d'angle :
MeasureInDegrees 0 en radian (défaut), 1 en degrés. Pour réutiliser une mesure d'angle en degrés, il faut la convertir en radian par un calcul.
DirectedAngles 0 entre 0 et π ou 180, 1 entre -π et +π ou -180 et +180 (défaut)

Enfin la taille de l'applet est définie dans la balise <applet> elle même :
width=xxx **height**=yyy

Le paramètre **codebase**="../jsp" définit l'emplacement où est stocké l'applet Java dans l'arborescence des fichiers du site.

Le reste (**code**="GSP.class" **archive**="jsp4.jar") est imposé par la spécification de JavaSketchpad.

Sommaire des entités

Point (NUMX, NUMY)
FixedPoint (NUMX, NUMY)
Midpoint (SEGM)
Point on object (STRAIGHT|CIRCLE|POLY, NUMB)
Intersect (STRAIGHT, STRAIGHT)
Intersect1 (STRAIGHT|CIRCLE, CIRCLE)
Intersect2 (STRAIGHT|CIRCLE, CIRCLE)
Segment (PNT, PNT)
Ray (PNT, PNT)
Line (PNT, PNT)
Perpendicular (STRAIGHT, PNT)
Parallel (STRAIGHT, PNT)
Polygon (PNT, PNT, PNT, ETC)
Circle (PNT, PNT)
Circle by radius (PNT, SEGM|DIST|CALC)
Circle interior (CIRCLE)
Reflection (OBJ, STRAIGHT)
Dilation (OBJ, PNT, NUMB)
Dilation/SegmentRatio (OBJ, PNT, SEGM, SEGM)
Dilation/3PtRatio (OBJ, PNT, PNT, PNT, PNT)
Dilation/MarkedRatio (OBJ, PNT, CALC|RATIO)
Rotation (OBJ, PNT, NUMB)
Rotation/MarkedAngle (OBJ, PNT, PNT, PNT, PNT)
Rotation/MeasuredAngle (OBJ, PNT, ANGLE|CALC)
Translation (OBJ, NUMB, NUMB)
VectorTranslation (OBJ, PNT, PNT)
Translation/FixedAngle/MarkedDistance (OBJ, DIST|CALC, NUMB)
Translation/MarkedAngle/FixedDistance (OBJ, ANGLE|CALC, NUMB)
Translation/MarkedAngle/MarkedDistance (OBJ, ANGLE|CALC, DIST|CALC)
Length (SEGM, NUMX, NUMY, STRING)
Angle (PNT, PNT, PNT, NUMX, NUMY, STRING)
Perimeter (POLY, NUMX, NUMY, STRING)
Circumference (CIRCLE|INTERIOR, NUMX, NUMY, STRING)
Radius (CIRCLE|INTERIOR, NUMX, NUMY, STRING)
Area (CIRCLE|INTERIOR|POLY, NUMX, NUMY, STRING)
Slope (STRAIGHT, NUMX, NUMY, STRING)
Distance (PNT, PNT, NUMX, NUMY, STRING)
Ratio/Segments (SEGM, SEGM, NUMX, NUMY, STRING)
Ratio/Points (PNT, PNT, PNT, NUMX, NUMY, STRING)
Calculate (NUMX, NUMY, STRING, EXPR), (MEASURE, ETC)
Parameter (NUMB, NUMX, NUMY, STRING)
ShowButton (NUMX, NUMY, STRING), (OBJ, ETC)
HideButton (NUMX, NUMY, STRING), (OBJ, ETC)
MoveButton (NUMX, NUMY, NUMB, STRING), (PNT, FREE, ETC2)
AnimateButton (NUMX, NUMY, STRING), (FREE, PATH, ETC2), (NUMB, ETC), (FLAG, ETC), (FLAG, ETC)
SimultaneousButton (NUMX, NUMY, STRING), (BUTT, ETC)
Locus (POINT|STRAIGHT, FREE, PATH, NUMB)
FixedText (NUMX, NUMY, STRING)
PeggedText (PNT, CAPT|MEASURE)
ConcatText (NUMX, NUMY), (CAPT|MEASURE, ETC)

Fonctions de JavaSketchpad non prises en compte par le compilateur

Diverses fonctions de JavaSketchpad sont complètement ignorées (= rejetées) par le compilateur.

Il est ainsi illégal d'utiliser les fonctions sur images :
Image, **ImageOnPoint**, **ImageBetweenPoints** ainsi que l'option d'affichage **image**('url'), ceci dans le cadre de l'utilisation en ligne pour des raisons de sécurité (pas d'url externes autorisées, pas d'images disponibles en local à priori).

Les fonctions sur systèmes de coordonnées **Origin&Unit UnitCircle AxisX AxisY PlotXY UnitPoint** ne sont pas disponibles, pour simplifier le compilateur, de même que les alias **DriverPoint** et **PolarTranslation**.

L'option **auto** est rejetée, pour imposer un démarrage manuel des applications, ainsi que les options de formatage de texte :
bold italic plain size(number**) font(**string**) justifyLeft justifyRight justifyCenter** (pour simplifier le compilateur) et l'option **suffix**(string) est rejetée pour la même raison.

Par contre un contrôle plus strict du typage des paramètres est effectué par le compilateur, rejetant des constructions qui passeraient dans JavaSketchpad avec un résultat douteux ou qui seraient tout bonnement ignorées.

Macros

Les macros et structures de contrôle n'existent pas dans la version actuelle du compilateur.
La syntaxe prévue pourrait être :

#define <nom> (type, type, …);
<corps de la macro>
#end

appel :
{id} nom (para, para, …);

Dans le corps de la macro les lignes sont ordinaires, avec uniquement les identifiants qui sont :
-En paramètre, #1, #2 … étant les paramètres formels 1,2,… remplacé par les paramètres réels au moment de l'expansion de la macro.
-En id, remplacés par $*id ou * est le nombre de fois où la macro a été appelée

La macro retourne comme objet le dernier crée.
Exemple :

```
#define Bissector (pnt, pnt);
{$s} Segment (#1,#2)[hidden];
{$m} Midpoint ($s)[hidden];
Perpendicular($s,$m);
#end
```

un appel :

```
{d} Bissector (A,B)[blue];
{e} Bissector (C,D)[red, thick];
```

génère alors :

```
{$1s} Segment(A,B)[hidden];
{$1m} Midpoint ($1s)[hidden];
{d} Perpendicular ($1s, $1m)[blue];
{$2s} Segment (C,D)[hidden];
{$2m} Midpoint ($2s)[hidden];
{e} Perpendicular($2s,$2m)[red, thick];
```

qui est traduit ensuite comme d'habitude
Le texte source des macros n'est pas transmis au résultat généré.

Structures de contrôle

Elle ne sont valables qu'à l'intérieur des macros

```
#if exp     exp = test sur valeur constante  ou type de paramètre (pas de test sur une variable)
            #if   #1=PNT  test si le paramètre #1 est un Point
            #if   #1=3      test si le paramètre #1 est le nombre 3 (directement, pas la valeur d'une
                            mesure, ceci est évidemment  imposible au compilateur.)
#else
#endif

#for  $i = liste de paramètres ou $i = valeur to n by step   valeur, n et step pouvant être des #p
#endfor
```

Copyright et licences

JavaSketchpad® est la propriété de Key Curriculum Press Inc. Son utilisation est définie par les accords de licence définis sur leur site.
http://www.keypress.com/sketchpad/javasketchpad/

Le compilateur est ma propriété personnelle. J'en autorise l'utilisation libre depuis mon site http://chephip.free.fr/ dans le but de générer vos propres dessins, publiés sur mon site ou pour votre usage personnel. Dans ce dernier cas, vous devrez vous procurer vous même JavaSketchpad auprès de Key Curriculum Press pour les visualiser.

Aucune garantie n'est assurée quant au bon fonctionnement du compilateur ou aux risques de perturbations et pertes de données de l'ordinateur sur lequel il est exécuté.

Version 1.0 (création) Jun 2006 Copyright Philippe Chevanne